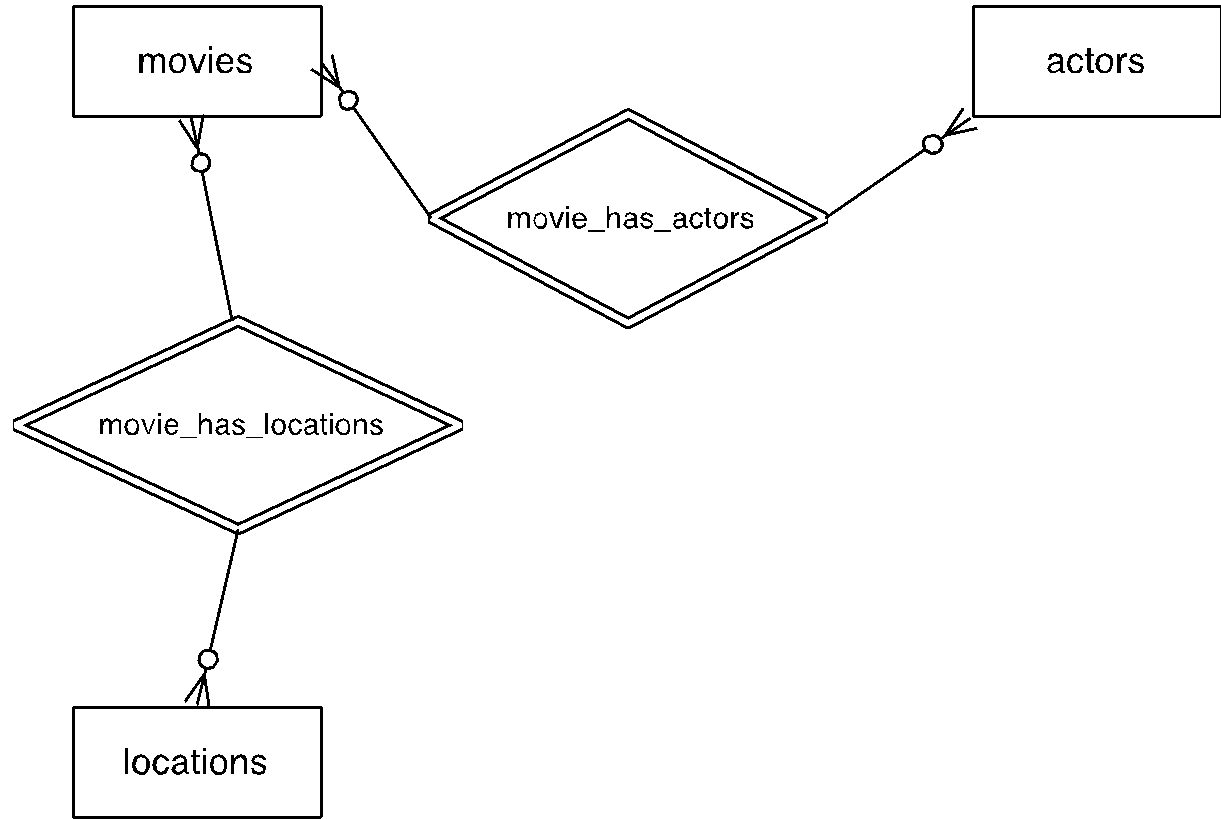
# INFSCI 2710 – Database Management – Spring 2018

## Homework 2 – More SQLs

1. Write SQL queries for each of the tasks using phpMyAdmin.
2. Follow the underscore convention when naming your database schema, entities and attributes.
3. Use the settings in lab session to connect to your MySQL and phpmyadmin
4. For all tasks, use the database that has been created by you. The database should have the same name as your Pitt username (the first part of your Pitt email)
5. Output your whole database (data + structure) as a SQL file. (So, I can import your database to my server through the SQL file)
6. **Attach your SQL statements + screen shot (the SQL execution result on your screen; INSERT/UPDATE/DELETE execution will show the effected rows or changes instead of return rows) below each question and save it as Word/PDF file**
7. Name the Word document (PDF) with your answers as ***YourPittID\_infsci2710\_homework2.docx (pdf)***. In other words, if your Pitt ID (first part of your Pitt email) is abc123, your submission file should be named ***abc123\_infsci2710\_homework2.docx (pdf)***
8. Submit your work (SQL file + Document) via CourseWeb.

Consider the relational database schema diagram below. It consists of 3 entities – movies, actors, and locations.



Because movies and actors have a many-to-many relationship and movies and locations have a many-to-many relationship, your schema would also have 2 junction tables – movies\_actors and movies\_locations. Each table’s logical structure is described below:

**Movies**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Primary Key** | **Data type** | **Is null?** | **Is auto-increment?** |
| movie\_id | yes | INT | no | yes |
| title | no | VARCHAR(200) | no | no |
| release\_date | no | DATETIME | no | no |
| plot\_description | no | VARCHAR(4000) | no | no |

**actors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Primary Key** | **Data type** | **Is null?** | **Is auto-increment?** |
| actor\_id | yes | INT | no | yes |
| first\_name | no | VARCHAR(100) | no | no |
| last\_name | no | VARCHAR(100) | no | no |
| birth\_date | no | DATETIME | no | no |
| biography | no | VARCHAR(1000) | no | no |

**locations**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Primary Key** | **Data type** | **Is null?** | **Is auto-increment?** |
| location\_id | yes | INT | no | yes |
| location\_name | no | VARCHAR(100) | no | no |
| street\_address | no | VARCHAR(150) | no | no |
| city | no | VARCHAR(100) | no | no |
| state | no | CHAR(2) | no | no |
| zip | no | VARCHAR(5) | no | no |

**movies\_actors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Primary Key** | **Data type** | **Is null?** | **Is auto-increment?** |
| movie\_id | no | INT | no | no |
| actor\_id | no | INT | no | no |

**movies\_locations**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field name** | **Primary Key** | **Data type** | **Is null?** | **Is auto-increment?** |
| movie\_id | no | INT | no | no |
| location\_id | no | INT | no | no |

**Task 1 (15 points):** In database **[your Pitt username]**, create the following entity tables:

1. movies
2. actors
3. locations

Each table’s logical structure should correspond to the descriptions provided in this assignment. Use CREATE TABLE statement.

**Task 2 (10 points)**: In your database, create the following junction tables:

1. movies\_actors
2. movies\_locations

Use CREATE TABLE statement to create junction tables. Make sure to create appropriate foreign keys – each table will have two foreign keys. Use ALTER TABLE statement to create foreign keys.

**Task 3 (15 points):** For each entity table, insert at least 5 rows using INSERT statement:

1. At least 5 movies in the **movies** table
2. At least 5 actors in the **actors** table
3. At least 5 locations in the **locations** table

You can make up your own data for the INSERT statements.

**Task 4 (10 points): For each junction table, create at least five relationships (insert at least five rows of proper IDs).**

**Task 5 (5 points): Write a SELECT statement to display top 2 actors sorted by actor’s last name.**

**Task 6 (5 points): Write a SELECT statement to display location name, street address, and city sorted by location name in descending order.**

**Task 7 (5 points): Write a SELECT statement to display movies released between two dates of your choice.**

**Task 8 (5 points): Write an UPDATE statement to update zip code for all locations to 15217**

**Task 9 (5 points): Write a statement that deletes one row of your choice from the actor's table. Be careful – make sure to write a correct WHERE clause.**

**Task 10 (5 Points): Write a SELECT statement to display movies which are played by any actor/actress who is a Gemini (or pick one Horoscope Sign you like)**

**Task 11 (5 points): Write a SELECT statement to display movies which the main cast (all actor/actress) has the highest average age.**

**Task 12 (5 points): Write a SELECT statement to display the Greatest Extra (http://www.imdb.com/title/tt3087298/) in your movie database, i.e., the actors who have appeared in the most movies.**

**Task 13 (5 Points): Write a SELECT statement to display the number of actor occurrence count of each city, i.e., how many actors ever appear in each city? [Output: locations.city, “number of actor occurrence count”]**

**Task 14 (5 Points): Write a SELECT statement to display an actor-to-actor network who ever act more than once in the same movie, i.e., to show a list of actor ID pair (ActorID1, ActorID2) that co-act more than one time.**

**\*Hint: for some questions, you may need to insert more data so that you can see the meaningful output.**